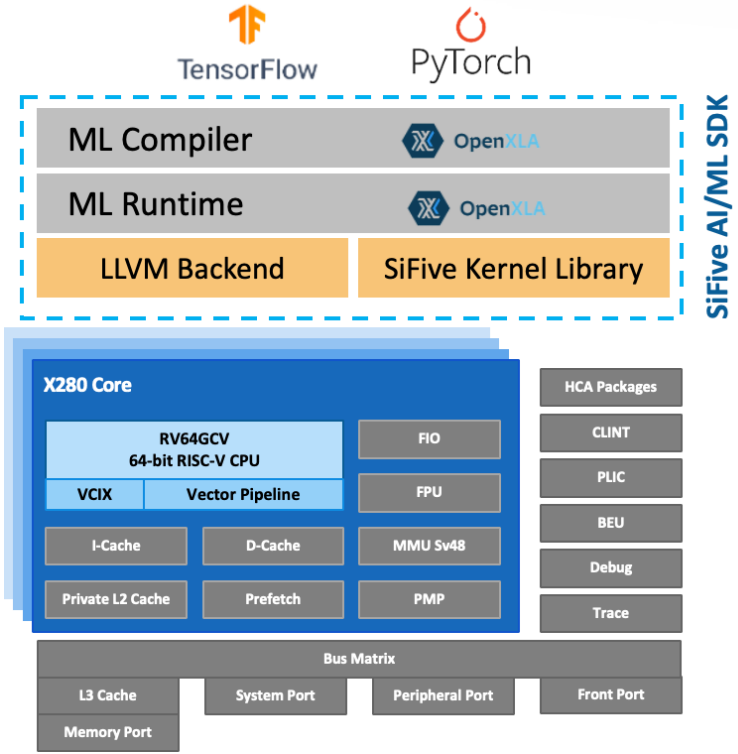
SiFive

Software Components and Methodology for Designing and Optimizing RISC-V ML Compilers

Hong-Rong Hsu

Pen Li

SiFive Intelligence is supported by AI/ML SDK with open-source ML Compiler and Runtime

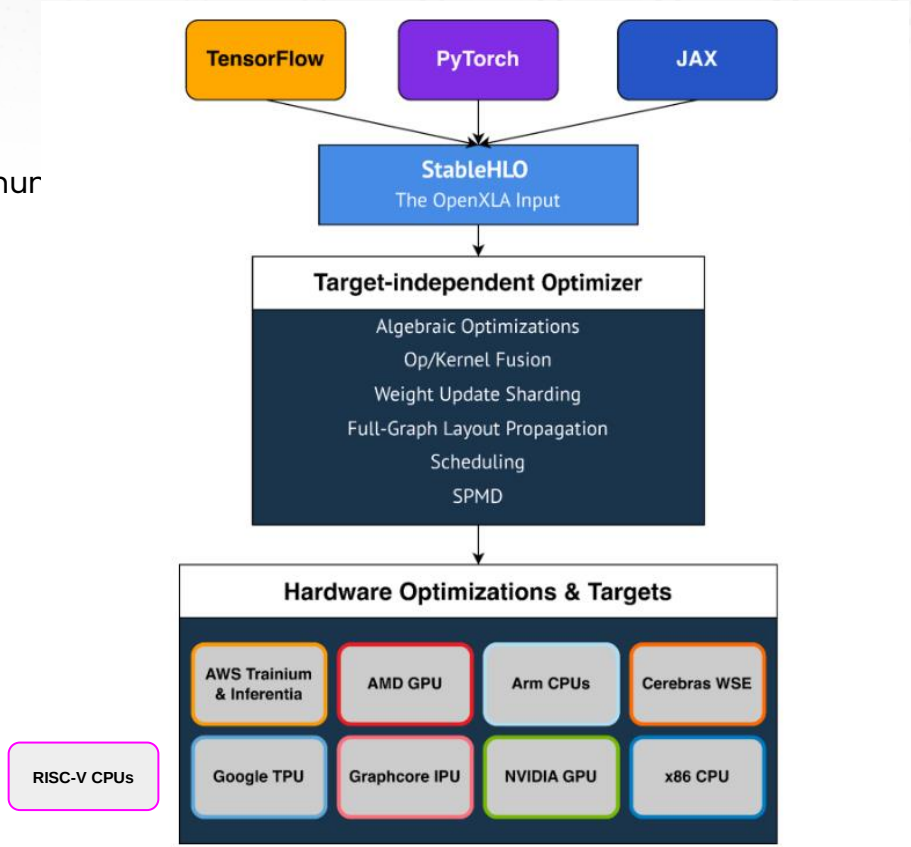


OpenXLA OpenXLA

- SiFive is one of the members and the only one company target RISC-V CPU in the OpenXLA commur

Alphabetical List

Org Name	Website	Join Date
Alibaba	alibaba.com	09-2022
Amazon Web Services	aws.amazon.com	09-2022
AMD	amd.com	09-2022
Anyscale (Ray)	anyscale.com	04-2023
Apple	apple.com	09-2022
Arm	arm.com	09-2022
Cerebras	cerebras.net	02-2023
Google	google.com	Founding
Graphcore	graphcore.ai	02-2023
Hugging Face	huggingface.co	02-2023
Intel	intel.com	09-2022
Meta (PyTorch)	about.meta.com	09-2022
NVIDIA	nvidia.com	09-2022
SiFive	sifive.com	02-2023



High-level OpenXLA compilation flow and architecture. Depicted optimizations, frameworks and hardware targets represent a select portion of what is available to developers through OpenXLA.

https://opensource.googleblog.com/2023/03/openxla-is-ready-to-accelerate-and-simplify-ml-development.html

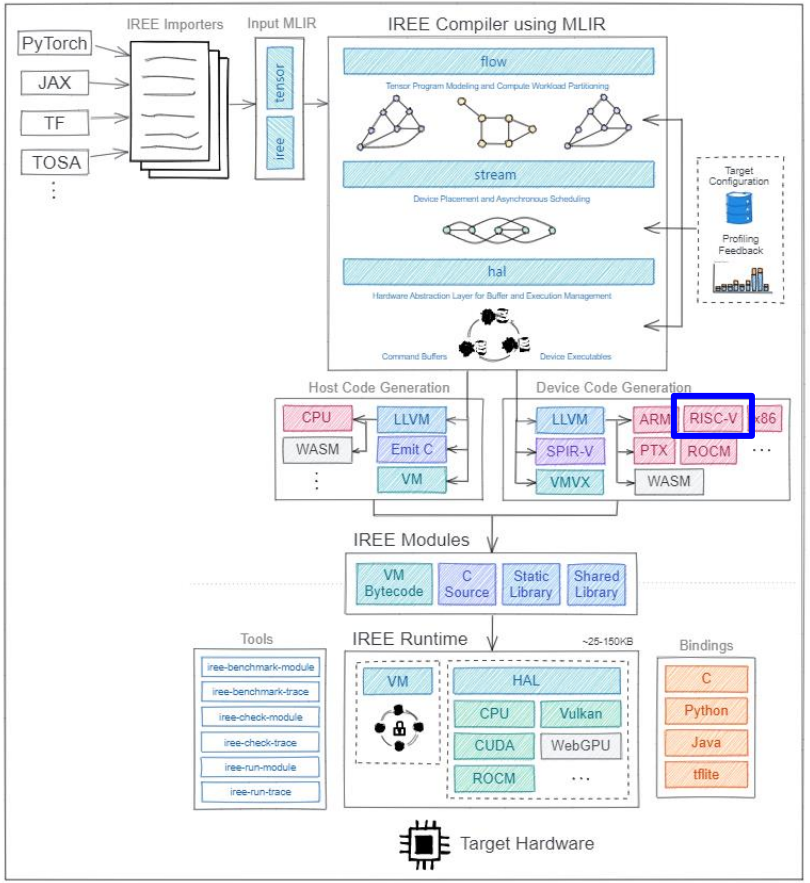
Pytorch model e2e Demo

Sentiment analysis Pytorch model(Hugging Face) -> OpenXLA/IREE -> X280 FPGA

```
root@sifive-fpga:~/yunh# cat /proc/cpuinfo
processor       : 0
hart            : 1
isa             : rv64imafdcv_zicsr_zifencei_zfh_zba_zbb_zvfh_xsfvfhbfmin_xsfvfnrclipxfqf_xsfvfwmaccqqq_xsfvqmaccqoq
mmu             : sv48
uarch           : sifive,bullet0

processor       : 1
hart            : 0
isa             : rv64imafdcv_zicsr_zifencei_zfh_zba_zbb_zvfh_xsfvfhbfmin_xsfvfnrclipxfqf_xsfvfwmaccqqq_xsfvqmaccqoq
mmu             : sv48
uarch           : sifive,bullet0

root@sifive-fpga:~/yunh# python3 demo.py
Demo model: MiniLM-L6-H384-uncased-sst2
Type a sentence to predict
That is a happy person
input sentence: That is a happy person
input sentence tensor: [101, 2008, 2003, 1037, 3407, 2711, 102, 0, 0, 0, 0, 0]
Positive: 0.9984435499542719
Negative: 0.0015564500457282428

Type a sentence to predict
you pass your input through the transformer model, then you have to apply the right pooling-operation on-top of the contextualized word embeddings
The sentence is too long, please try a short one (token <= 10)

Type a sentence to predict
The quick brown fox jumps over the lazy dog.
input sentence: The quick brown fox jumps over the lazy dog.
input sentence tensor: [101, 1996, 4248, 2829, 4419, 14523, 2058, 1996, 13971, 3899, 1012, 102]
Positive: 0.02506056148210878
Negative: 0.9749394385178911

Type a sentence to predict
```

OpenXLA / IREE

- Is an MLIR-based end-to-end compiler and runtime that's part of the open source, community-driven OpenXLA echosystem
- Mainly apply AOT compilation solution
- LLVM CPU lowering path: through Vector and LLVM dialects



Question:

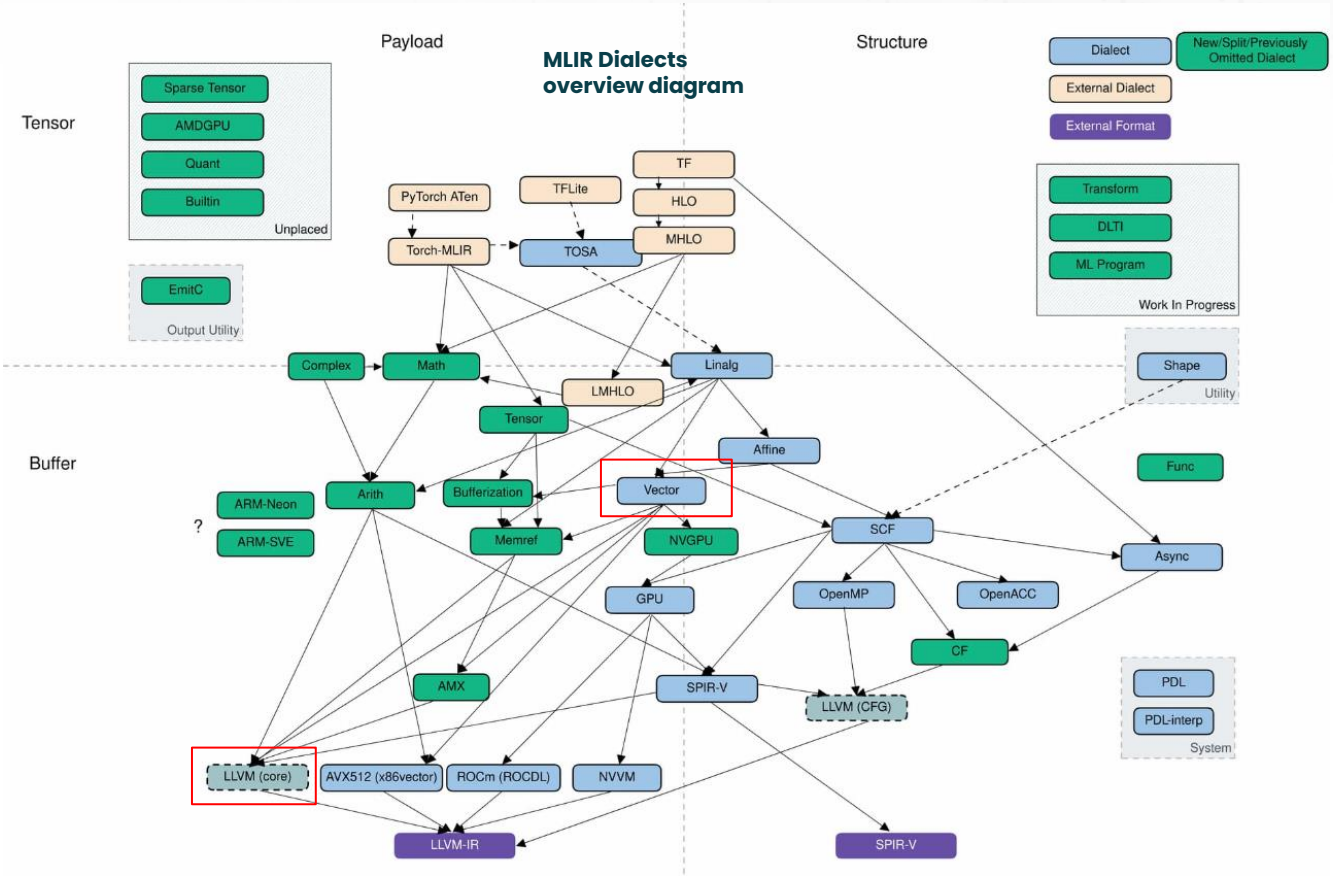
So just hook RISC-V LLVM backend up to ML compiler, everything is done?

Matmul RVV code-gen

```
...
vle32.v     v29,(a0)
addi        a0,s1,-128
vle32.v     v30,(a5)
addi        a5,s1,-64
vle32.v     v31,(a4)
addi        a4,s1,64
vle32.v     v8,(a0)
vle32.v     v9,(a3)
vle32.v     v10,(a5)
vle32.v     v11,(s1)
vle32.v     v12,(a4)
vrgather.vi v13,v29,0
vfmadd.vv   v13,v8,v25
vrgather.vi v25,v30,0
vfmadd.vv   v25,v8,v26
vrgather.vi v26,v9,0
vfmadd.vv   v26,v8,v28
vrgather.vi v28,v31,0
vfmadd.vv   v28,v8,v27
vrgather.vi v27,v29,1
vfmadd.vv   v27,v10,v13
vrgather.vi v8,v30,1
vfmadd.vv   v8,v10,v25
vrgather.vi v25,v9,1
vfmadd.vv   v25,v10,v26
vrgather.vi v26,v31,1
vfmadd.vv   v26,v10,v28
vrgather.vi v28,v29,2
vfmadd.vv   v28,v11,v27
vrgather.vi v27,v30,2
vfmadd.vv   v27,v11,v8
vrgather.vi v8,v9,2
vfmadd.vv   v8,v11,v25
vrgather.vi v10,v31,2
vfmadd.vv   v10,v11,v26
vrgather.vi v25,v29,3
...
```

MLIR Compiler lowering analysis

- Vector and LLVM are key dialects for RVV code-gen



https://discourse.llvm.org/t/rfc-updated-mlir-dialect-overview-diagram/64266

After adding an optimization pass

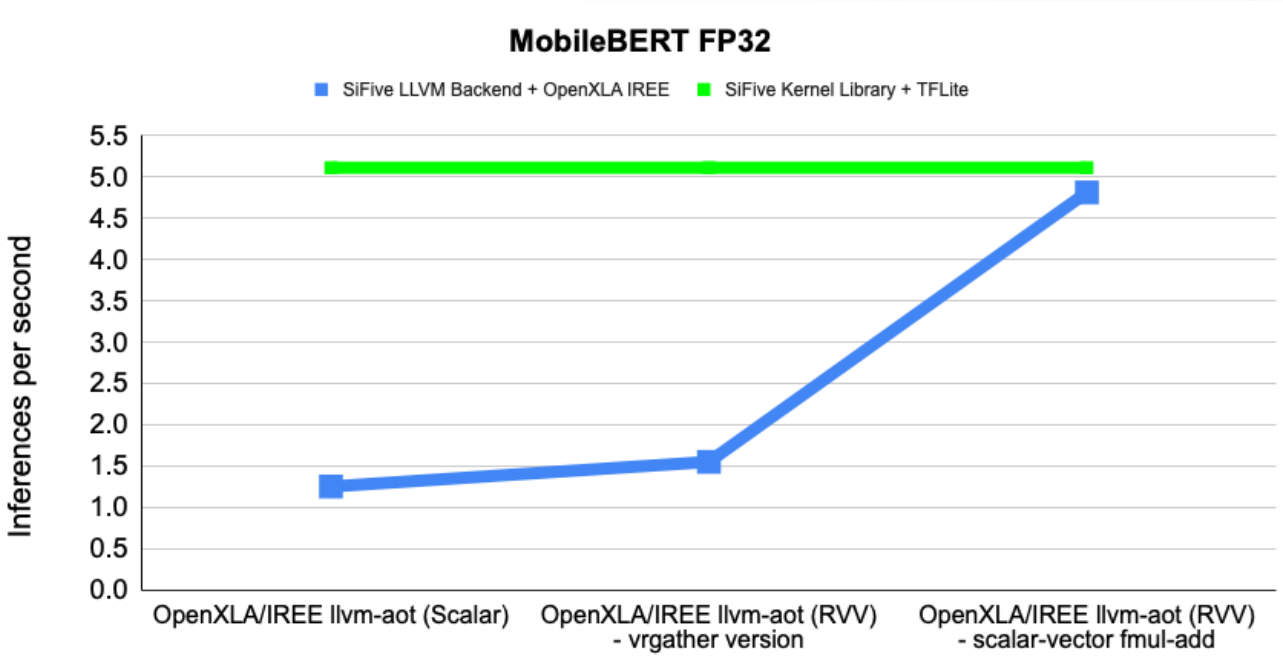
```
...
vle32.v      v29,(a0)
addi         a0,s1,-128
vle32.v      v30,(a5)
addi         a5,s1,-64
vle32.v      v31,(a4)
addi         a4,s1,64
vle32.v      v8,(a0)
vle32.v      v9,(a3)
vle32.v      v10,(a5)
vle32.v      v11,(s1)
vle32.v      v12,(a4)
vrgather.vi  v13,v29,0
vfmadd.vv    v13,v8,v25
vrgather.vi  v25,v30,0
vfmadd.vv    v25,v8,v26
vrgather.vi  v26,v9,0
vfmadd.vv    v26,v8,v28
vrgather.vi  v28,v31,0
vfmadd.vv    v28,v8,v27
vrgather.vi  v27,v29,1
vfmadd.vv    v27,v10,v13
vrgather.vi  v8,v30,1
vfmadd.vv    v8,v10,v25
vrgather.vi  v25,v9,1
vfmadd.vv    v25,v10,v26
vrgather.vi  v26,v31,1
vfmadd.vv    v26,v10,v28
vrgather.vi  v28,v29,2
vfmadd.vv    v28,v11,v27
vrgather.vi  v27,v30,2
vfmadd.vv    v27,v11,v8
vrgather.vi  v8,v9,2
vfmadd.vv    v8,v11,v25
vrgather.vi  v10,v31,2
vfmadd.vv    v10,v11,v26
vrgather.vi  v25,v29,3
```

```
...
vle32.v v7,(a0)
flw     fs11,0(s7)
vle32.v v6,(a1)
flw     ft0,0(a4)
vle32.v v5,(a3)
vfmacc.vf       v28,ft2,v2
flw     ft2,4(a2)
vfmacc.vf       v27,fs4,v2
flw     ft3,8(a2)
vfmacc.vf       v26,fs5,v2
flw     ft4,12(a2)
vfmacc.vf       v25,fs3,v2
flw     ft5,16(a2)
vfmacc.vf       v10,fa7,v2
flw     fs5,20(a2)
vfmacc.vf       v9,fs0,v2
flw     fs4,24(a2)
vfmacc.vf       v8,fs1,v2
flw     fs3,28(a2)
vfmacc.vf       v31,fs2,v2
flw     fs2,32(a2)
vfmacc.vf       v30,fs6,v2
flw     fs1,36(a2)
vfmacc.vf       v29,fs7,v2
flw     fs0,40(a2)
vfmacc.vf       v16,fs8,v2
flw     fs7,44(a2)
vfmacc.vf       v15,fs9,v2
flw     fa7,48(a2)
vfmacc.vf       v14,fs10,v2
flw     fs9,52(a2)
vfmacc.vf       v13,fs11,v2
flw     fs8,56(a2)
flw     fs10,60(a2)
vfmacc.vf       v12,ft0,v2
vfmacc.vf       v11,ft1,v2
...
```

Matmul-f32 performance get **3.1x** times improvement (see later pages for e2e improvement)

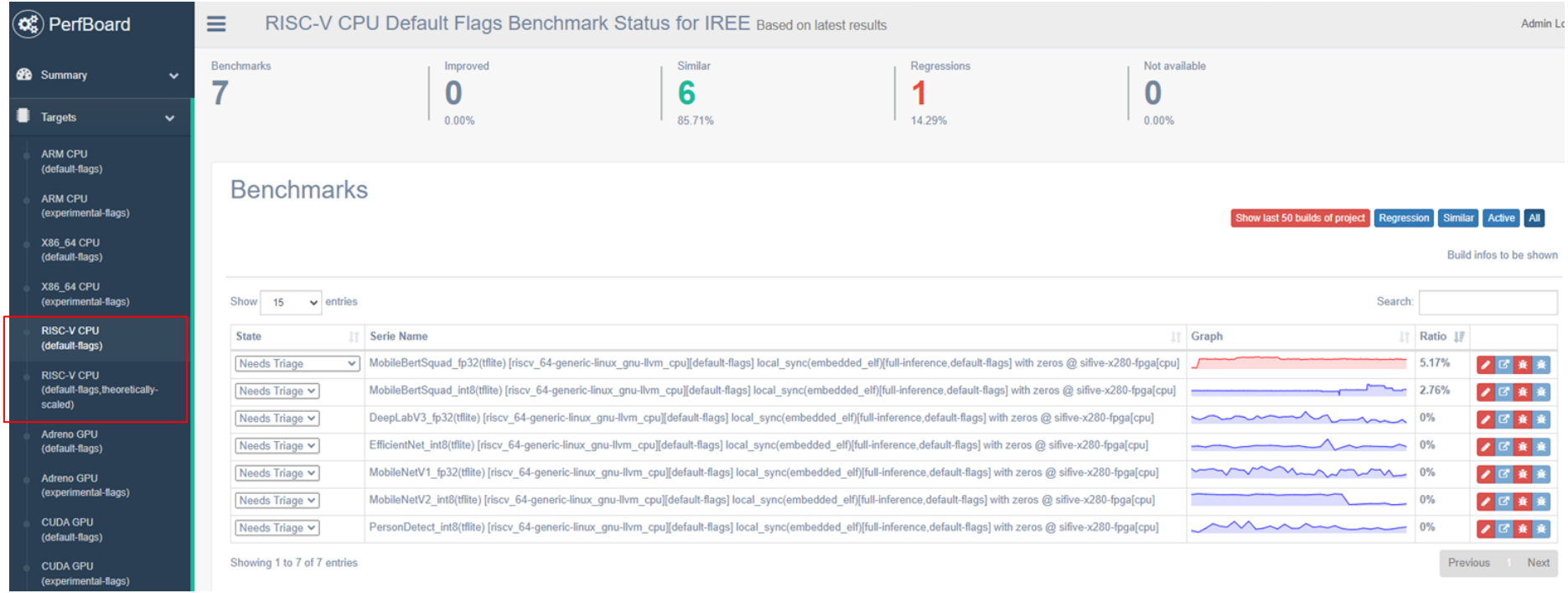
All of the opts are in the LLVM mainstream!

Inference Execution Performance



e2e Performance on IREE's dashboard

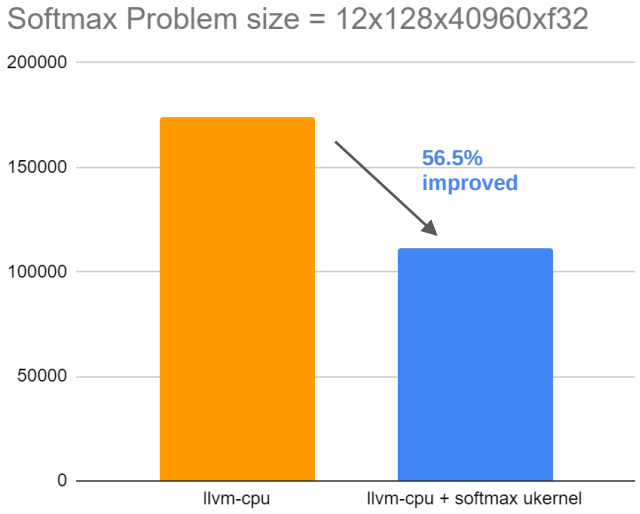
Check out the RVV e2e performance of the various AI/ML models online
https://perf.iree.dev/IREE/default-riscv-cpu-series



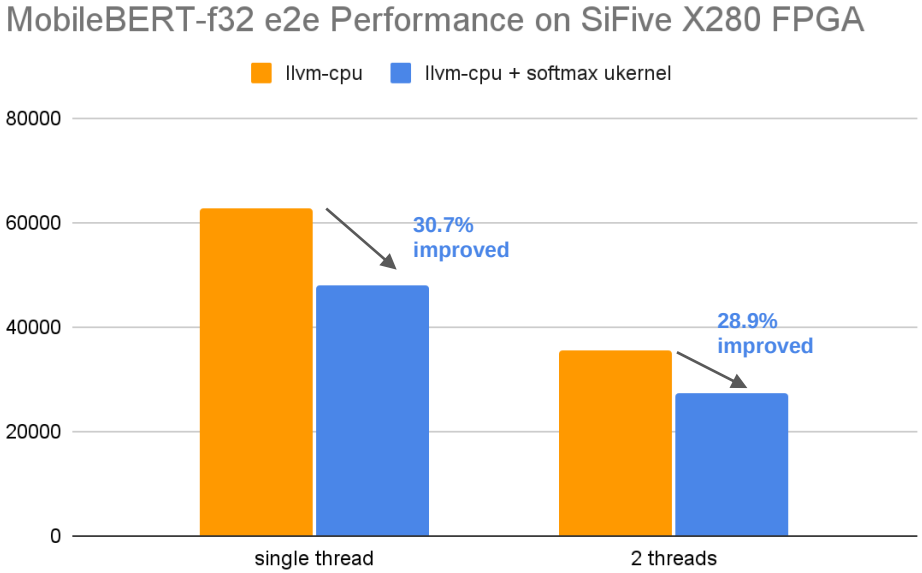
U-Kernel Integration

OpenXLA/IREE llvm-cpu + u-kernel path

- SiFive RVV Softmax u-kernel integration
- Benchmark on SiFive X280 FPGA



iree/tests/e2e/regression/softmax_large.mlir



U-Kernel Integration

llvm-cpu + u-kernel path

- OpenXLA/IREE allows people to vectorize and code-gen RVV(RISC-V Vector extension)

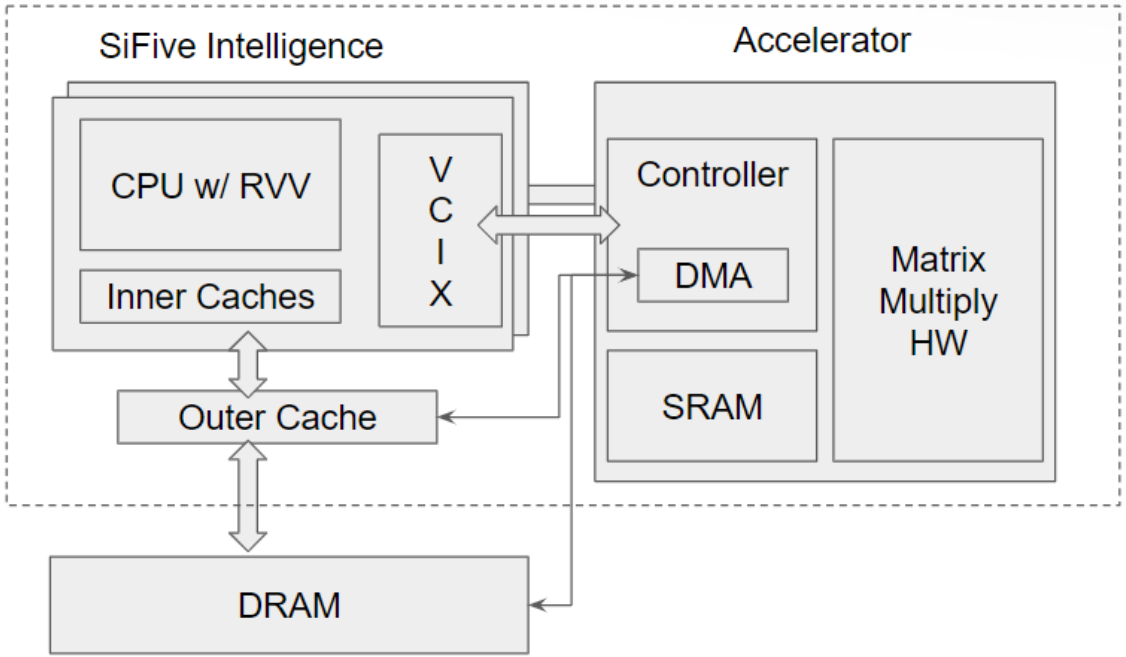
- For some ops not code-gen well in current compiler
  - ops with custom instructions
  - ops with micro-architecture specific optimization

- U-kernel can help complement the performance
- U-kernel can be a reference for compile to code-gen better

CPU + DLA Example

Scalable from Edge to Data Center & ADAS

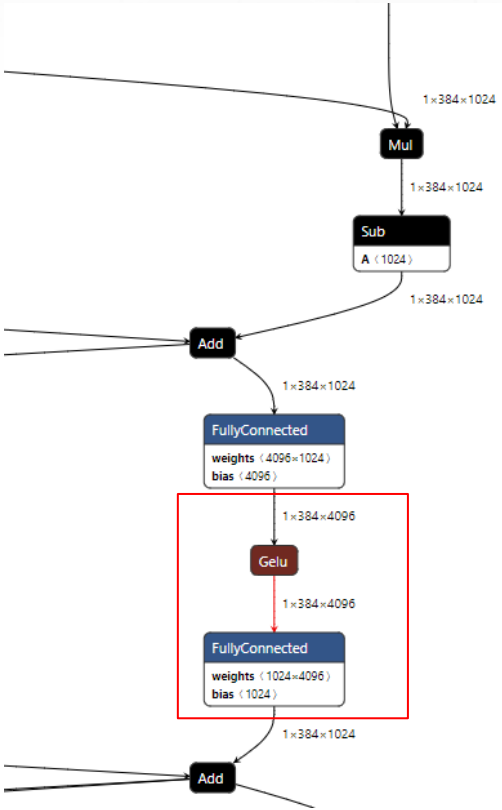
OpenXLA +



Plan to RFC the VCIX dialect on MLIR

- Why VCIX?
  - Tight coupling between CPU and XPU/co-processor
  - CPU and co-processor runs in parallel
  - One program with scalar, vector, and co-processor instructions interleaved
- VCIX dialect allows people to
  - manage the control & optimization flow in MLIR's world

```
scf.for %arg0 = %c0 to %c384 step %c1 {
  scf.for %arg1 = %c0 to %c4096 step %vl {
    // RVV process GELU
    // Push %vl data from vector registers to co-processor via VCIX
  }
  scf.if (%arg0 == %matmul_tile_m_size) {
    // Send command to trigger co-processor for matmul via VCIX
  }
}
// Pop the matmul results back via VCIX
```



BERT-large usage example

Empowering innovators

www.sifive.com